Design Methodology 1:

가상 프로토타이핑 기법을 이용한 설계자산의 기능검증

기안도 ㈜다이나릿시스템 연구소장 (adki@dynalith.com)

I. 요약

설계자산의 통합으로 구성하는 시스템집적반도체의 설계에 있어서 각 설계자산의 기능은 완벽하게 검증되어야 한다. 이를 위해서는 설계자산의 기능검증환경이 매우 효과적으로 구축되어야 하며, 본 고에서는 이를 위한 방법으로 능동적이고 지능적인 시험환경에 대해 소개하고, 설계자산의 기능 검증에 대해 분석하고 정리한다. 그리고 가상프로토타이핑 기법에 대해 소개하고 이를 적용한 설계 예를 통해 다양한 응용에 대해 가상프로토타이핑이 어떻게 구성되며 어떤 효과가 있는지 정리한다.

II. 능동적이고 지능적인 시험환경의 필요성

일반적으로 채택되고 있는 반도체 칩 개발과정은 로직 설계, 게이트 수준 설계, 배치 및 배선 그리고 타이밍 분석, 반도체 칩 제조 단계를 따른다. 각 단계에서 수행되는 디버깅 과정의 결과에 따라 앞선 과정을 반복해서 수행하기도 한다. 특히 주목할 점은 각 단계에서 기능을 검증하기 위해 시뮬레이션을 기본으로 사용하고 있다는 것이다. 여기서 시뮬레이션이란 순수소프트웨어 시뮬레이터 프로그램을 이용하여 설계를 가상적으로 동작시켜 정확하게 응답하는지를 확인하는 것이다.

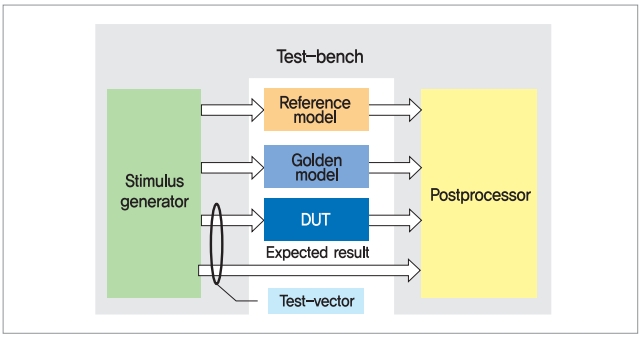
반도체 칩을 목표로 하는 디지털 하드웨어 설계의 경우 특별한 언어가 필요하며, 이 언어는 일종의 병렬프로그래밍 언어이기 때문에 특별한 프로그램이 있어야 모델링 된 병렬 현상을 확인할 수 있다. Verilog, VHDL, SystemC 등이 대표적인 경우이며, 이들을 위한 시뮬레이터가 있다.

외부와 정보전달이 없는 하드웨어 블록은 설계의 대상이 될 수 없으며, 이것이 시사하는 바는 설계된 블록의 동작을 확인하기 위해서는 입력을 인가하고 출력을 분석하는 것이 필요하다는 것이다. 분석된 결과가 기대와 다른 경우 원인을 찾기 위해서는 블록 내부의 동작을 확인하는 것도 필요하다.

가장 단순한 방법은 설계한 블록의 입력 신호와 출력 신호 그리고 중요한 내부 자원의 동작 신호를 파형으로 분석하는 것이다. 이것이 가장 확실한 방법이지만, 신호파형을 보고 그 의미를 이해하고, 기대하는 파형이 어떤 것인지 유추할 수 있어야만 가능한 방법이다. 또한 입출력 신호의 수가 많거나 입출력 신호에 실리는 정보가 중요한 의미를 갖는 경우는 파형을 분석하여 설계한 블록의 기능을 검증하는 것이 불가능한 일이 될 수 있다.

복잡한 기능을 갖는 설계 블록의 동작을 시뮬레이션 하기 위해서는 역시 매우 복잡한 시험각본이 필요하다. 만약 시뮬레이션 과정에서 설계 오류를 발견하여 설계를 수정하게 되면, 이전의 모든 시험각본을 다시 시뮬레이션 할 필요가 생긴다. 이러한 상황에서 앞서 언급한 신호파형을 단순하게 분석하는 방법을 사용하는 것은 매우 어리석은 일이며, 큰 비용과 긴 시간이 요하게 됨으로 공학적인 측면에서 불가능한 것이 될 수 있다.

따라서 입력에 대한 출력을 자동으로 분석하여 정확한 동작인지 여부를 확인하는 능동적이며 지능적인 시험환경이 필요하다. 〈그림 1〉과 같이 DUT(Design Under Test)를 설계함에 있어 입력을 생성하는 블록(Stimulus generator: SG), 출력을 분석하는 블록(Postprocessor: PP)으로 구성된 시험환경(Test-bench)이 필요하다. 특히 SG는 입력과 그 입력에 대한 기대 출력(expected result)으로 구성되는 시험벡터(Test-vector)를 생성하고 PP는 출력과 기대 출력을 비교하여 DUT가 정확하게 동작하였는지 확인하는 구성이 가능하다.



〈그림 1〉 시뮬레이션 환경의 구성

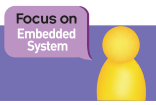
그러나 DUT의 기능이 복잡하고 입출력의 정보가 많은 경우 SG가 기대 출력을 생성하는 것 또한 어려운 일이므로, 골든 모델(Golden model) 또는 참고 모델(Reference model)을 동시에 사용하는 것이 더욱 효과적인 방법이 될 것이다. 여기서 골든 모델은 해당 블록을 설계하는 과정 중 이전 단계에서 기능이 검증된 블록을 의미하고, 참고 모델은 해당 블록의 규격을 충실하게 반영한 완벽한 모델이라고 할 수 있다.

예로써, FFT(Fast Fourier Transformation) 블록을 설계할 경우 부동소수점(floating-point) 연산과 자료형을 사용한 알고리즘 수준의 모델이 참고 모델이고, 고정소수점(fixed-point) 연산과 자료형을 사용한 비트수준에서 정확도를 갖는 모델이 골든 모델이다. 또는 Matlab/Simulink와 같은 이미 검증된 함수를 지원하는 추가적인 소프트웨어 패키지를 시뮬레이션에 접목하여 참고 모델로 활용할 수 있다. 이때 DUT와 골든 모델의 비교는 DUT가 제대로 설계되어 동작하는지를 확인하기 위함이 목적이고, DUT와 참고 모델의 비교는 DUT의 결과가 어느 정도 수준으로 정확한지를 확인하기 위함이 목적이다.

멀티미디어 정보(음향과 영상)와 신호처리를 필요로 하는 정보를 다루는 설계들의 경우 신호 파형으로 검증할 수 있는 부분은 극히 일부분이며, 대부분의 경우는 입출력 정보의 내용을 분석하는 것이 중요하다. 따라서 보다 실감나고 감성적인 측면까지 동원할 수 있는 시험 환경이 필요하다. 예로써, MPEG2/4, JPEG 등과 같은 기능에 연관된 블록의 경우 화상 출력을 통해 기능을 검증하는 것이 더욱 효과적일 것이다. 즉, 〈그림 1〉과 같은 환경에서 후처리 블록에 영상출력, 음향출력, 데이터 프로세싱 등과 같은 기능을 구현할 경우 매우 효과적인 기능 검증 환경이 될 것이다.

Ⅲ. 설계자산의 기능검증

특정 기능 블록을 구현한 설계자산(IP: Intellectual Property)은 개별 블록으로 설계되는 것이지만 궁극적으로 보다 큰 시스템에 통합되어 동작해야 되므로 기능검증 측면에서도 이러한 점이 반영되어야 한다. 비록 단위 기능을 구현하는 것이 목적이지만 다른 기능 블록들과 연동하여서 정확한 기능으로 그리고 충분한 성능으로 동작하는 것을 명확하고 객관적으로 증명해야 한다.



설계자산의 기능을 검증하기 위해 필요한 시험들을 다음과 같이 구분해 볼 수 있다.

- 호환성 시험 (Compliance testing): 규격을 충실하게 반영하였는지 시험한다. 특히 표준에 관련된 경우라면 충분하게 모든 기능이 구현되었는지 반드시 점검해야 된다. 보통의 경우 점검 리스트나 호환성 시험 패키지들이 있고, 표준화 기구에서 인증서를 발급한다.
- 모서리 시험 (Corner case testing): 매우 복잡한 경우와 특별한 상황에 대해 시험한다. 이는 설계자산이 제대로 동작하지 않을 가능성이 높은 경우를 시험하는 것이다.
- 무작위 시험 (Random testing): 호환성 시험과 모서리 시험은 계산된 상황을 적용하여 시험하는 것인 반면, 무작위 시험은 설계자가 미처 고려하지 못한 경우를 시험함으로써 예상하지 못한 문제를 찾아낼 수 있다.
- 실제 응용 시험 (Real code testing): 해당 설계자산이 사용될 응용에 직접 적용하여 시험하는 것으로 규격을 잘못 이해한 경우나 잘못된 규격을 기준으로 설계하거나 시험한 경우를 찾아낼 수 있다.
- 확인 시험 (Regression testing): 설계자산의 시험과정에서 발견된 문제를 해결한 후, 새로운 설계자산이 이전 시험각본들에 대해 여전히 문제없이 동작하는지를 시험한다. 이는 특정 문제해결이 다른 새로운 문제를 파생시킬 수 있기 때문에 반드시 필요한 시험이다.

설계자산에 대한 이러한 기능 검증을 하기 위해서 다음과 같은 검증 장비와 환경들이 사용된다.

- 시뮬레이션 (Simulation): 사건구동 시뮬레이터를 통해 기능을 검증하는 것으로, RTL(Register-Transfer-Level)의 경우 효과적이다. 그러나 설계자산의 규모가 크거나 게이트 수준 설계의 경우 리셋 처리와 같은 초기상황을 점검하는 정도에 만족해야 되며 확인 시험, 실제 응용 시험, 무작위 시험 등에는 시뮬레이션 성능이 충분하지 않기 때문에 적용하기 어렵다.
- 시험환경 자동화 도구 (Testbench automation tools): Verilog나 VHDL의 기본 기능에 시험 벡터를 생성하고 결과를 분석하는 등의 기능을 체계적이고 용이하게 추가할 수 있도록 한다. 무작위 시험이나 시험 정도 분석 등 시험환경 구축에 도움이 되는 기능을 제공하며 Vera, E, SCV(SystemC Verification Library) 등이 있다.
- 코드 유효시험 분석 도구 (Code coverage tools): 정적인 검증 기법(Link Checking ; formal model checking)이 아닌 동적인 검증 기법의 경우 유효시험의 범위는 시험환경에서 생성한 시험각본에 따라 결정된다. 따라서 어느 정도 시험이 되었는지 확인하는 것이 필요하며, 코드 유효시험 분석 도구는 정량적으로 분석한 결과를 제공하고, VeriSure, VHDLCover, VeriCov, CoverMeter 등이 있다.
- 하드웨어 모델 (Hardware modeling): 실소자와 소프트웨어 시뮬레이터 사이의 인터페이스를 통해 시뮬레이터에서 실소자의 입출력을 사용할 수 있도록 한다. 이러한 환경은 설계자산에 해당하는 실소자가 있는 경우 설계한 설계자산의 기능을 시뮬레이터에서 비교할 수 있으므로 매우 높은 신뢰성을 갖고 시험할 수 있다. 이러한 환경을 이용할 경우 가상프로토타이핑이 가능하다.
- 에뮬레이션 (Emulation): 특별한 하드웨어를 이용하여 설계자산을 하드웨어로 실행시키는 방법이며 비교적 규모가 작은 설계사산에 적용하기에는 적합하지 않다. 그러나 최근에는 하드웨어 가속과 에뮬레이션이 모두 지원하는 기술과 중소규모 제품이 있기 때문에 설계자산 검증에도 활용해 볼 수 있다.
- 프로토타이핑 (Prototyping): FPGA와 같은 프로토타이핑 소자를 이용하여 실제 하드웨어 시스템을 만드는 것이며 실제 응용 코드를 준 실시간으로 동작시킬 수 있기 때문에 매우 효과적인 방법이다. 그러나 제작하는데 소요되는 비용과 노력이 크다는 단점이 있고, 주변 블록들도 모두 하드웨어로 구현해야 된다는 문제가 있다.

하드웨어로 구현될 설계자산은 활용 측면에서 다음과 같은 기준으로 설계되어야 한다.

- 일반화된 문제(General problem)를 다룰 수 있도록 설계: 다른 응용에 용이하게 적용할 수 있어야 한다.
- 다양한 구현기술(Technology)에 적용될 수 있도록 설계: Soft IP의 경우는 합성 스크립트가 여러 합성기와 하드웨어 라이브러리에 쉽게 적용될 수 있도록 준비되어야 하고, Hard IP의 경우는 새로운 구현기술에 적용하는 구체적이고 상세한 방법이 준비되어야 한다.
- 다양한 시뮬레이터에 적용될 수 있도록 설계: 여러 시뮬레이션 스크립트가 여러 로직 시뮬레이터에 적용될 수 있도록 준비되어야 하고, 다양한 설계 언어와 연동이 될 수 있도록 준비되어야 한다.
- 다양한 검증 환경에 적용될 수 있도록 설계: 보다 큰 시스템에 포함되어 검증이 가능하도록 준비되어야 하며, 동시에 독자적인 검증이 가능하도록 준비되어야 한다.

- 충분한 설명자료: 앞서 설명된 기준에 대한 상세하고 충분한 설명과 적용 가능한 환경 그리고 사용의 제한 등에 대한 정확한 설명이 준비되어야 한다.

이상의 기준을 만족시키려면 설계자산 자체를 잘 설계하는 것 만으로는 부족하다는 것을 알 수 있다. 즉, 설계자산이란 특정 시스템 설계에 국한하여 블록을 설계하는 것이 아니라 보다 일반적인 용법으로 다른 응용에도 사용할 수 있도록 설계하는 것이며, 이는 설계자의 관점보다는 이용자/활용자의 관점에서 설계되고 활용환경이 동시에 준비되어야 한다.

Ⅳ. 가상프로토타이핑

실제적으로 구현하지 않고 마치 실제로 존재하는 것과 유사한 또는 동일하게 동작하는 모델을 만드는 것을 프로토타이핑이라 한다면, 넓은 의미에서 시뮬레이션 모델도 프로토타이핑의 한 예가 될 수 있다. 그러나 능동적이고 보다 실제적인 입출력 현상을 구현한 경우를 좁은 의미에서 프로토타이핑이라 할 수 있고, FPGA(Field Programmable Gate Array)와 같은 재설정이 가능한 하드웨어 소자를 이용한 하드웨어 기반 프로토타이핑 보드가 이것에 속할 것이다. 로직 설계 단계와 게이트 수준 설계 과정 사이에 기능 검증의 추가적인 단계로 활용된다. 하드웨어 프로토타이핑은 동작 속도와 실감적인 측면에서 매우 효과적인 기능 검증 방법이지만 높은 비용과 큰 노력이 수반된다는 것과 다른 응용으로 재활용하기 어렵다는 단점이 있다.

하드웨어 프로토타이핑이 갖는 단점은 FPGA 보드를 설계하고 제작하고 디버깅해야 한다는 것에 기인한다. 특히 관심의 대상인 블록이 아닌 시험 환경까지 모두 하드웨어로 구현해야 된다는 점은 매우 불합리하다고 할 수 있다. 따라서 하드웨어 제작을 최소화하면서 능동적이고 보다 실제적인 입출력 현상을 구현할 수 있다면 매우 효과적일 것이다.

호스트 컴퓨터에서 수행되는 프로그램과 FPGA를 연동하는 기술과 제품인 iPROVE가 있으므로 이것을 이용한다면 하드웨어 제작 없이 프로토타이핑이 가능할 것이며, 이러한 것을 가상 프로토타이핑(Virtual prototyping)이라 한다. 가상프로토타이핑은 넓은 의미에서 현실성 있는 입출력 기능이 지원되는 소프트웨어적인 시스템 모델을 포함하며, 좁은 의미에서는 FPGA와 소프트웨어 모델을 연동하여 만든 시스템 모델을 말한다.

Ⅴ. 가상프로토타이핑 기법의 적용사례들

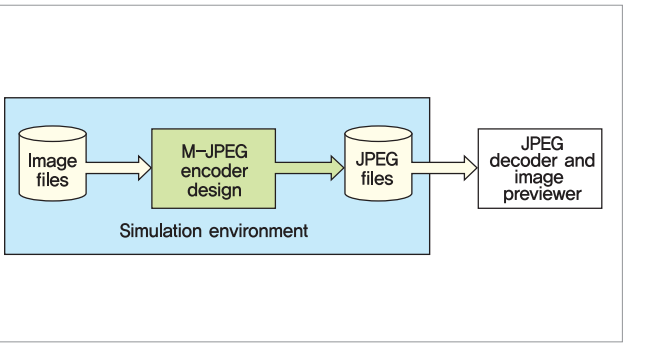
앞에서 설명한 가상프로토타이핑 기법을 적용한 예제들을 통해 어떻게 적용되고 어떤 효과가 있는지 살펴본다.

1. 동영상정보를 다루는 M-JPEG 설계

멀티미디어 정보 중 영상정보의 경우 시각적인 감성 효과를 직접 점검하는 것이 필요하고, 특히 동영상의 경우 충분한 프레임 속도로 자연스러운 영상 재현이 되는지를 확인하는 것이 중요하다.

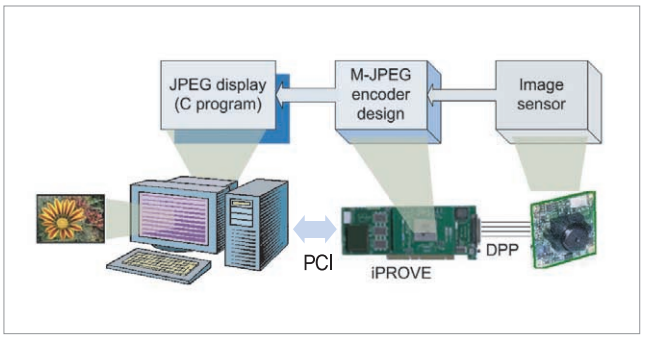
M-JPEG 블록은 이미지 센스로부터 동영상정보를 입력 받아 JPEG 형식으로 압축하는 기능을 한다. 따라서 M-JPEG 블록의 시험을 위해서는 M-JPEG 블록, 동영상 이미지 생성 블록, M-JPEG 디코딩 블록 그리고 동영상 이미지 출력 블록 등이 필요하다.

보통의 경우 〈그림 2〉와 같이 동영상 이미지 생성 블록은 미리 저장해 둔 이미지 파일을 입력으로 사용하고, M-JPEG동영상 이미지 출력 블록은 생성된 정보를 파일에 저장하도록 한 후, 시뮬레이션이 끝나고, JPEG 파일들을 디코딩하여 이미지 출력 프로그램으로 확인한다. 이런 기능검증환경은 개별 이미지 프레임을 정확하고 면밀하게 검사하기 위해 꼭 필요하다. 그러나 시뮬레이션과 이미지 정보의 분석 과정이 분리되어 있기 때문에 여러 가지로 불편한 점이 많다.

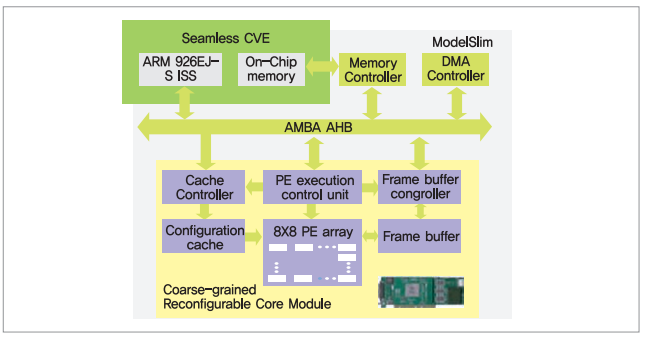


〈그림 2〉 M-JPEG 시험환경의 일례

〈그림 2〉에서 입력과 출력 부분을 시뮬레이터와 연동되는 프로그램으로 대체할 경우 검증생산성은 매우 높아진다. 그러나 M-JPEG과 같이 동영상을 다루는 경우 시뮬레이션으로는 충분한 프레임 속도를 얻기 어려우므로 한계가 있다. 따라서 M-JPEG 블록의 설계가 어느 정도 진행되어서 FPGA에 맵핑 할 수 있는 단계가 되면 가상프로토타이핑 기술을 적용해 볼 수 있다.



〈그림 3〉 M-JPEG encoding에 적용된 가상프로토타이핑



〈그림 4〉 재구성가능 코어 설계에 적용된 가상프로토타이핑

〈그림 3〉은 Motion JPEG 엔코딩 블록에 가상프로토타이핑 기법을 적용한 예이다.

이미지 센서를 FPGA에 연결하고, FPGA에 맵핑 된 M-JPEG 블록이 생성한 JPEG 정보를 호스트컴퓨터의 JPEG 디코딩 및 이미지 출력 프로그램으로 출력하게 함으로써 보다 실감나고 고속으로 전체 시스템을 시뮬레이션 할 수 있다. 이 예는 MagnaChip(구 Hynix) 사의 System IC Institute에서 다이나릿시스템의 iPROVE를 적용하여 구현한 것이다. 이 경우 ModelSim을 이용한 순수 소프트웨어 시뮬레이션의 경우에는 한 개 프레임을 엔코딩하는데 약 2시간이 소요되었고, iPROVE를 적용하여 ModelSim과 연동한 경우에는 약 11분이 소요되었다. 반면 C 프로그램을 이용한 전송수준 시뮬레이션의 경우 초당 15 프레임을 엔코딩하는 결과를 얻었다.

M-JPEG의 예에서 주목할 점은 다음과 같다.

- 기능을 검증할 설계자산을 FPGA에 맵핑하여 게이트 수준에서 기능을 검증한다.
- FPGA와 실소자인 이미지 센서를 연동한다.
- 고급 프로그래밍 언어인 C를 설계자산의 기능검증에 사용한다.
- 영상 출력과 같은 가상적인 입출력 장치를 프로그램으로 연동한다.

2. 프로세싱코어를 활용하는 설계

시스템집적반도체의 경우 프로세싱코어를 포함하게 되고 이러한 시스템에 포함되는 설계자산은 시뮬레이션에서 프로세싱코어와 연동되는 환경에서 기능이 검증되어야 한다. 특히 설계자산이 매우 복잡한 프로그래밍 환경을 요할 경우 단순한 시험벡터로 기능을 점검하기 어렵다.

〈그림 4〉에 서울대학교 설계자동화연구실에서 재구성가능 코어의 기능 검증에 iPROVE를 활용하여 구성한 가상프로토타이핑 환경을 보였다. 재구성가능 코어는 응용에 따라 기능의 재구성이 가능한 일종의 프로세서이므로 다른 프로세싱 코어로 제어하는 기능이 필요하며, 여기에서는 ARM9 프로세서를 이용하였다. 〈그림 4〉에서 노란색으로 채워진 아래쪽 큰 블록이 iPROVE에 맵핑된 재구성가능 코어이고, ARM9은 Seamless CVE와 ModelSim을 통해 iPROVE와 연동되었다. 이 경우 DCT 응용을 수행하였을 때 단순한 시뮬레이션(ModelSim)보다 약 9배, Integer Transform 응용을 수행하였을 때 단순한 시뮬레이션보다는 7배 성능 향상을 보였다.

재구성가능 코어에 적용된 가상프로토타이핑의 예에서 주목할 점은 다음과 같다.

- 기능을 검증할 설계자산을 FPGA에 맵핑하여 게이트 수준에서 기능을 검증한다.
- 프로세싱 코어 모델링으로 Seamless CVE와 같은 HW/SW 동시협조시뮬레이터를 사용한다.
- AMBA 시스템은 ModelSim과 같은 HDL 시뮬레이터를 사용한다.

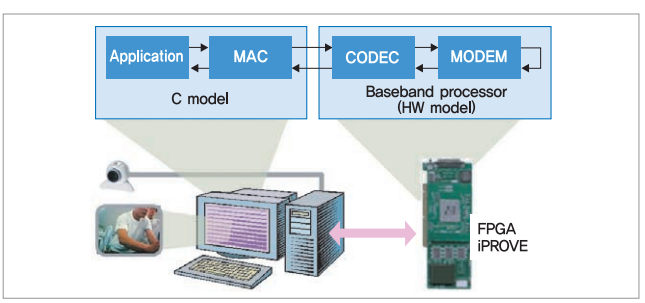
3. 통신 응용에 활용한 예제 설계

통신에 관련된 설계는 점점 그 중요도가 높아지고 있다. 특히 무선통신에 관련된 설계는 더욱 복잡하고 성능개선에 대한 요구가 높기 때문에 가능한 짧은 시간에 개발해야 되는 상황이다. 통신응용에 적용되는 설계는 매우 다양한 평가 지표가 있고, 이들 지표에 대한 분석을 위해서는 다양한 시험벡터와 시험모드에 따른 시뮬레이션이 필요하다. 아울러 실동작 상황에서 어떤 품질로 어느 정도의 성능으로 동작할 것인지 미리 확인하는 것이 필요하다.

〈그림 5〉는 충북대학교 CCNS 연구실에서 WLAN(IEEE

802.11a)의 기저밴드 프로세서에 대한 검증에 iPROVE를 적용한 예를 보였다. WLAN의 경우 무선주파수 부분이 필요하고, MAC과 응용 부분이 모두 있어야 제대로 된 기능검증이 가능하다. 그러나 설계단계에 이들 모두를 동시에 하나의 환경에 구축하는 것은 쉽지 않다. 따라서 〈그림 5〉와 같이 RF 부분은 생략하고 출력을 입력으로 연결한 루프백(loopbak)을 사용하며, 상위 기능블록들은 고급언어인 C 프로그램으로 대체하였다.

WLAN 설계를 이용하는 응용으로 영상정보를 송수신하는 예를 설정하고, 영상입력으로 PC 카메라를 사용하고, 루프백되어 돌아온 영상정보는 이미지 출력 프로그램으로 출력하는 구성을 하였다. 이러한 구성으로 WLAN 기저밴드 프로세서를 위한 FPGA 프로토타이핑 보드를 만드는 노력 없이 해당 설계자산에 대한 FGPA 수준 기능검증이 가능하였고, 동시에 실시간에 가까운 성능으로 동작시연이 가능하였다.



〈그림 5〉 WLAN용 기저밴드 프로세서에 적용된 가상프로토타이핑

WLAN의 예에서 주목할 점은 다음과 같다.

- 기능을 검증할 설계자산을 FPGA에 맵핑하여 게이트 수준에서 기능을 검증한다.
- 고급 프로그래밍 언어인 C를 설계자산의 기능검증에 사용한다.
- PC 카메라와 영상 출력과 같은 가상적인 입출력 장치를 프로그램으로 연동한다.
- 설계자산을 이용하는 응용에 대해 실감나는 정도의 성능으로 시연하였다.

VI. 종합 및 결론

반도체 칩 설계를 통해 구현할 기능들이 복잡해짐에 따라 기능검증에 대한 부담이 더욱 높아졌다. 특히 시스템집적반도체의 경우 전체를 한몫에 설계하지 않고 확보 가능한 설계자산을 최대한 활용하고, 새로 설계할 기능블록은 설계자산으로 개발하여 전체를 통합하는 과정을 통해 설계하는 것이 일반적이 되었다.

이러한 상황에서 설계자산의 기능검증이 기본적이면서도 무엇보다 중요한 것이 되었기 때문에 각 블록설계자들은 보다 신뢰도 높은 설계자산 개발을 위한 방법을 필요로 한다. 본 고에서는 시험환경에 대해 주목하였고, 특히 능동적이며 지능적인 시험환경이 설계자산 개발에 왜 중요한 것인지 어떤 역할을 하는 것인지에 대해 소개하였다. 특히 골든모델과 참고모델을 활용할 경우 설계자산의 기능검증에 높은 생산성을 확보할 수 있음을 설명하였다. 또한 설계자산의 기능검증에 적용되는 시험을 목적에 따라 구분하여 각각이 어떤 것이며 무슨 목적인지를 설명하였다. 아울러 설계자산의 기능검증에 필요한 환경과 장비에 대해 정리하였고, 이어서 설계자산 개발에 있어 기준이 되어야 할 점들에 대해 정리하였다.

또한 시스템집적반도체에 포함될 설계자산의 경우 칩 내부 네트워크를 통해 프로세싱 코어의 프로그램 제어로 기능을 발휘하는 설계자산을 어떤 시험환경을 구성함으로써 보다 용이한 기능검증이 가능한지에 대해 설명하였다. 그리고 영상 음향 등 멀티미디어 정보를 입출력으로 하는 설계자산의 경우 가상프로토타이핑 기법을 통해 보다 효과적이며 실감나는 시험환경을 구축할 수 있음을 설명하였고, 동영상, 프로세싱 코어 제어를 필요로 하는 설계, 통신 등에 관련된 예제를 통해 가상프로토타이핑이 어떻게 적용되었고 어떤 결과를 얻었는지를 정리하였다.

시스템집적반도체가 주류를 이루는 반도체설계 특히 설계자산 개발에 있어 기능검증은 가장 기초적이면서도 중요한 것이므로 이를 위한 개발환경이 지능적이고 능동적으로 구축되어야 하며 가상프로토타이핑 기법을 적용할 경우 매우 효과적이고 고성능으로 실감나는 기능검증 환경이 만들어 질 수 있다.

[참고문헌]

[1] M. Keating and P. Bricaud, Reuse Methodology Manual for System-on-a-chip Design, Chapter 7. Macro Verification Guidelines, 2nd Ed., KAP, 2001.
[2] iPROVE User Manual, Dynalith Systems, 2005. (www.dynalith.com)
[3] ㈜하이닉스반도체, 휴대전화용 저전력 동영상 압축 CODEC 개발 최종결과 보고서, 2003.12, 한양대학교 CAD & 통신회로 연구실
[4] Y. Kim, M. Kiemb, C. Park, J. Jung, K. Choi, "Resource sharing and pipelining in coarse-grained reconfigurable architecture for domain-specific optimization", Design, Automation and Test in Europe, pp.12-17, Mar. 2005.
[5] T. Batsuuri, T.-I. Yun, Y.-S. Cho, K.-R. Cho, "Implementation and verification of MAC processor for WLAN on a SOC platform", ISOCC-2004.